PREAD User Interface

Hydrologic Engineering Center
Functions, Macros and Screens

User's Manual

March 1995

Hydrologic Engineering Center
U.S. Army Corps of Engineers
609 Second Street
Davis, CA 95616-4687
(916)756-1104

PREAD
Table of Contents

Chapter 1

1. Introduction

The incorporation of the PREAD user interface in an interactive or batch executed program can greatly enhance its ease of use. PREAD allows a program to conveniently take input from the keyboard as would normally occur in an interactive environment, but also allows several input alternatives.

Input alternatives available to any program using PREAD are:

1) Function references,
2) Macro procedures, and
3) Screen selection input.

Function references, macro and screen procedures may be utilized by interactive users at any kind of terminal. PC users may use a mouse to select items from a screen.

1.1 Function References

The function capability allows any ASCII character to be redefined to mean a string of zero or more ASCII characters. For example, the dollar sign "`$`" could be redefined to be the string "`PLOT`" or the pound sign "#" could be redefined to be the string "`THIS IS A TEST LINE`". Then each use of the $ or `#` characters would produce the redefined equivalent.

The function capability can be a significant asset for frequently typed strings entered at the keyboard. Function references can also be nested, so that one function reference may reference other functions. Functions may also be referenced by input lines from macros as discussed later. Function definitions are preserved in the function file.

1.2 Macro Procedures

Macro procedures are sets of recurring lines of input. If, for example, an interactive program requires 5 or 6 lines of input to create a certain plot, the 5 or 6 lines could be stored in a macro. When the plot is desired, the user runs the macro to produce the plot. The execution of a macro can make using an interactive program considerably more pleasant for repetitive uses. Lines stored in the macro can contain function references. Since macros can accept parameters at execution time, a macro with a function reference can produce 2 different plots if the function reference is redefined between its executions. Macros can also be nested. Macro definitions are preserved in the macro file.

1.3 Screen Selection

Screen selection allows the execution of complex programs by novice users, as well as by experienced users. When desired, the user is presented a screen of options from which to select. All entries by the user must come from the list of valid responses. A valid response causes the program to execute one or more operations. The user is prevented from making erroneous requests, and can only do what the selection screen permits. This is very useful for controlling access to high level programs where the user is unfamiliar with the program, or how to use it. Screen definitions are preserved in the screen file.

1.4 Support Capabilities

Each of the above capabilities is controlled by a separate file that describes the functions, macros, and screens available to the user. These files are normal text files that can be edited by the user to change the function definitions, macro procedures, or screens . The format of each of these files is given in the appendices of this document.

The user can define or redefine functions directly without the need to edit the appropriate text file. Similarly, the user can have PREAD create a macro as the steps are executed by the program. Screens can only be created or changed by use of a text editor. Optionally, commands can be echoed to the user terminal as they are executed. This can be helpful when function references are used as input or lines are retrieved from a macro file.

PREAD can also keep a log of all lines entered at the terminal, run from a macro, or selected from a screen. This feature can help reconstruct critical terminal sequences. Input lines are stored in the log file.

1.5 Extended Capabilities

PREAD extended capabilities require special implementation for each computer system used. The CHAIN feature allows the user to interrupt the execution of the program at any program input location and begin execution of any other program.   When the user has completed execution of the second program, the user can then return to the first program continuing execution where the interruption first occurred.

The JCL feature allows the user to move into job control at any point in the execution of a program and resume execution when JCL activities are complete. Caution must be exercised while in job control not to destroy files needed to resume execution of the interrupted program.

1.6 PREAD Commands

All lines that contain the command character in the first character position of a line are assumed to be commands to PREAD. The default command character is an exclamation point "!". A minus sign " - " following the command character and preceding the command will cause appropriate commands to take on their opposite meaning. (eg., `!-echo`) An equal sign "=" following the command character and preceding the command will cause appropriate commands to display current definitions. (eg., `!=function`)

Commands may be abbreviated to two or more characters.

Some commands accept option characters. Options are single characters separated from the command by a period "." (e.g., `!WAIT.X 10`). PREAD commands and options are case insensitive; that is, they may be entered in any combination of lower and upper case. Arguments may or may not be case sensitive depending on their usage.

Syntax:

```
!command [.option]  [arguments,...]
```

Examples:

```
!Teach  #  String
!wait.x  10
!Run  myMac
```

(This page intentionally left blank)

Chapter 2

Commands

2.1 Name: CHAIN

Use: `!CHAIN program name`

Description: The chain command allows any other program to be executed from the current host program. When the new program is terminated the execution of the original program is resumed where it was interrupted.

Example:

```
!CHAIN  DSSUTL
!chain  dsplay macfile=dsp123 dssfile=mydss
```

2.2 Name: COED

Use: `!COED  edit file name`

Description: The COED command allows the current host program to be interrupted to edit a file. Upon termination of the edit, control will revert to the host program.

Example:

`!COED MYDATA` - Edit the MYDATA file.
`!coed dspmac` - Edit the dspmac macro file.

2.3 Name: DISPLAY / LIST

Use: `!DI  [list file name] [first line of file to be displayed]`

Description: The DISPLAY and LIST commands are identical. They each show lines from a text file. If the command is given without a filename, the next series of lines from the file will be displayed. See the examples below.

Options: T - Text only will be output suppressing frame line normally shown at beginning of each page of listing.

Example:

`!LIST   MYTEXT`	Display file MYTEXT
`!list.t myfile 200`	Display file MYFILE beginning with Line 200 without header
`!LIST`	Display next series of lines
`!-Display`	Display previous series of lines

2.4 Name: ECHO

Use: `!ECHO`

Description: The line being processed by PREAD may be echoed to the screen. This is often of interest when the user wishes to see the lines of a macro being executed.

Example:

`!-ECHO`	- Turn echo off (if previously turned on)
`!echo`	- Turn echo on (if previously turned off)

2.5 Name: EDIT

Use: `!EDIT  macro-name`

Description: The edit command allows macros to be created, changed, or deleted on-line. The required parameter is the name of the macro to be edited. If the macro does not already exist it will be created. Only simple (I) insert, (D) delete, (R) replace, (N) next, (P) print, and (F) finish commands may be used. Major macro editing activities should be performed by exiting the program and using a normal text editor.

Example:

```
!EDIT PLOTIT
!edit bigmac
```

2.6 Name: FIND

Use: `!FIND   [file name] find string`

Description: The FIND command will find a string in a file. The file to be used may be specified, or will be the last referenced DISPLAY/LIST file. A FIND without a string will show the next occurrence of the string in the file.

Example:

```
!FIND MYOUTPUT SUMMARY
!find ERROR
!FIND
```

2.7 Name: FUNCTION

Use: `!FUNCTION`

Description: Turns on or off function mode, or shows current function definitions. The default at program initiation is OFF.

Example:

`!FUNCTION` Turn function mode on (any character that is defined in the function file will be replaced by its function value without the need for a function shift . . . . character).

`!-FUNCTION` Turn function mode off (function characters will be treated as normal characters unless preceded by the function shift character).

`!=FUNCTION` Display all defined functions (all currently defined function characters will be displayed with their definition).

If function mode is off (the default) a function reference may be forced by preceding the function reference with the function shift character. The default function shift character is the caret "^" . The function shift character is defined at the beginning of the function file and may be changed there by a text editor. It is generally recommended that the function mode be left in the OFF state and all desired function requests use the function shift character. For example, with automatic function replacement OFF if the pound sign, " # ", has been defined as the string "`123`", the expression "`ABC#DEF^#GHI`" would be expanded to be "`ABC#DEF123GHI`".

2.8 Name: HARDCOPY

Use: `!HARDCOPY`

Description: The hardcopy command will generate the code sequence required to trigger a screen copy for a Tektronix or compatible device.

Example:

```
!Hardcopy
```

2.9 Name: IF/ELSEIF/ELSE/ENDIF

Use:
```
!IF ( 'string1' .EQ. 'string2' ) THEN
!ELSEIF ( 'string3' .EQ. 'string4' ) THEN
!ELSE
!ENDIF
```

Description: The macro file may contain conditional execution lines governed by `IF`, ELSEIF, ELSE, and ENDIF statements. The conditional statements may be nested. Each IF sequence may be followed by any number of optional ELSEIF conditions, not more than one optional ELSE condition, and MUST be terminated by an ENDIF statement. Each of the expressions must be a character string. The string may be a function reference (e.g., `^a, ^M`), a parameter passed into the macro, a literal string (e.g., '`FIRST`', '`Bone Creek`'), or combinations thereof. The only logical tests permitted on the strings are equal "`.EQ.`" or not equal "`.NE.`".

Example:

```
!IF ( '^a' .EQ. 'FIRST' ) THEN
!RUN FIRSTONE
!ELSEIF ( '^a' .EQ. 'SECOND' ) THEN
!RUN SECONONE
!ELSEIF ( '^a' .NE. '999' ) THEN
!RUN ANYONE
!ELSE
!RUN EVERYONE
!ENDIF


!IF ( '$PARM' .NE. 'FLOW' ) THEN
!IF ( '$LOC' .EQ. 'BLACK BROOK' ) THEN
!Run Special $LOC
!ELSE
!Run Normal $LOC $PARM
!ENDIF
!ENDIF
```

2.10 Name: JCL

Use: `!JCL  [system command]`

Description: The JCL command allows the user to interrupt the current host program and return to the operating system. To resume execution of the interrupted program enter EXIT.

Example:

```
!JCL
PRINT MYFILE
EXIT
```

2.11 Name: KBLINE

Use: `!KBLINE  [function character]`

Description: The KBLINE command is used to interrupt a macro to allow the next line of input to be entered from the keyboard. After entry the currently executing macro resumes execution. If the command contains an optional function character, the next line entered from the keyboard will be used to define that function character.

Options:

`H` -	Hold moving to new line and hold displaying prompt.
`P` -	Prompt even when H option in effect.
`S` -	Screen response last selected will be used as the input.

Example:

`!KBLINE`	(Next line from keyboard will be past to program)
`!KBLINE.H $`	(Next line from keyboard will be stored as $, no prompt or newline will be issued)
`!kbline.s #`	(Last selection line from a screen will be stored in #)

2.12 Name: LEARN

Use: `!LEARN  new macro name`

Description: Creates a macro by learning (remembering) a series of entries made by the user. Macros may also be created by editing the macro file. Creating a macro by editing the macro file is the most common method to use.

Example:

`!learn abc`	(Begin saving lines in macro ABC. All subsequent input lines will be included in macro ABC)
`!-LEARN`	(End saving lines in current macro)

2.13 Name: LOG

Use: `!LOG`

Description: This command controls the logging of lines to the log file. All lines that are past to the host program may be written to a log file for future reference. The log command may enable or disable this feature.

Example:

`!Log`	- Enable log file (if previously desabled)
`!-LOG`	- Disable log file (if previously enabled)

2.14 Name: PAGE

Use: `!PAGE  [page type]`

Description: The page command will generate the code sequence required to trigger a new page (screen clear) for a Tektronix or compatible device. A parameter of GRAPHICS will cause a screen clear of the graphics area. A parameter of PRINTER will generate a formfeed character.

Example:

```
!PAGE
!PAGE GRAPHICS
!page printer
```

2.15 Name: PAUSE

Use: `!PAUSE`

Description: Causes a macro to pause until a carriage return (CR) is entered from the keyboard. This allows a macro to contain a series of plots, or other steps, and allow the user to control when its execution will proceed.

Options:

`H` - Hold moving to new line and hold displaying prompt.
`P` - Prompt even when H option in effect.

Example:

```
!PAUSE
!pause.hp
```

2.16 Name: PRINT

Use: `!PRINT  [message to be printed]`

Description: The print command allows a message to be displayed to the user regardless of the echo status. If the .C option is used, the message will be written at the cursor location specified and exactly three fields must follow the command. The cursor line and column fields must be numeric values. If the .C option is used, the message must be enclosed in quotes. If the numeric values are unsigned, absolute screen coordinates are used. If they are signed, relative movements from current location are made. With the .C option, the cursor remains at last position after message. Zero length messages are permitted.

Option:

`C` - Cursor positioning used.

Example:

```
!PRINT -- Press <RETURN> to continue --
!print.c 10,20 "Message prints at line 10, column 20"
!PRINT.C 11,20 "Message prints at line 11, column 20"
!print.c +2,-20 "Message prints 2 lines down and 20 columns left"
```

2.17 Name: RUN

Use: `!RUN  macro name`

Description: Runs a macro, displays names of macros, displays the contents of a macro. Macros may be passed parameters at execution time. See Appendix B for examples of macros defined in a file and parameter passing conventions.

Example:

`!run one`	(Execute macro named ONE)
`!=RUN`	(Display names of all defined macros)
`!=RUN TWO`	(Display contents of macro TWO)
`!run PLOTIT RED STAGE OBS`	(Execute macro named PLOTIT, passing three parameters RED, STAGE, and OBS.)

2.18 Name: SCREEN

Use: `!SCREEN  screen name`

Description: Invokes the selection screen from which the user may choose a valid response. The screen may provide narrative descriptions of desired actions which may be selected by the simple entry of a one or more character strings. Only valid responses are accepted. PC users may optionally use a mouse to make selections.

Options:

`V` - Force screen as visible regardless of other controls.
`I` - Force screen as invisible regardless of other controls.
(only a prompt will be displayed)
`D` - Delete saved PUF screen name. This option is only valid on the PC.

Example:

`!screen main`	
`!=SCREEN`	List names of available screens
`!screen.v helpscn`	Only screen prompt line will be displayed
`!SCREEN BLUERES`	
`!Scr  fast|`	(MS DOS only) vertical bar as end character of screen name causes screen to be saved for faster display on subsequent uses.
`!screen.d  faster|`	(MS DOS only) delete "faster|" screen from saved area so that changes in imbedded function characters can change screen display

2.19 Name: SET/GET

Use:
```
!SET STATUS keyword OFF | ON
!GET STATUS keyword function character
```

Description: The `SET/GET` commands allow getting and setting various control information. `STATUS` information on current settings as displayed in the `STATUS` command may be stored in functions with the `GET` command. Valid keywords for `STATUS` requests are: `FUNCTION`, `ECHO`, `LOG`, `MENU`, `AUTOSCREEN`, `MACRO`, `BREAK`, and `LEARN`. Changing `MACRO` or `LEARN` by use of the `SET` command is not valid. The `SET` command can be used to set values of the same control flags to either `ON` or `OFF`.

Example:

```
!SET STATUS FUNCTION OFF
!GET STATUS ECHO &
!set STATUS ECHO &
!SET TERMINAL TYPE 4107
```

2.20 Name: STATUS

Use: `!STATUS`

Description: The status command displays the current status of each of the capabilities of PREAD.

Example:

```
!status
```

2.21 Name: TEACH

Use: `!TEACH  function string`

Description: Defines, re-defines or removes a function character. Functions of zero length may be defined, whereby all occurrences are replaced with nothing. The maximum length of an individual function string is 30 characters on the PC and 80 characters on other systems.

Example:

`!teach @ Test`	(Define the at sign @ to mean "Test")
`!-TEACH #`	(Remove the definition for the pound # sign)
`!TEACH %`	(Define zero length function for %)

2.22 Name: WAIT

Use: `!WAIT  number of seconds`

Description: The wait command forces the macro to wait the specified number of seconds before proceeding. This is useful when a continuous display of information is made and time is needed to allow the screen to be read as the macro executes.

Option: `X` - Exits all macros running if a user enters an "X" character during the wait.

Example:

```
!WAIT 15
!wait.x 10
```

2.23 Name: * Command

Use: `!*  [comment]`

Description: An `*` as the first character following the command character signifies that the line is a comment. Comments will not be passed to the program.

Example:

`!*` This line is a comment

2.24 Name: ? Command

Use: `!?`

Description: The entry of a ? will display a table of allowable PREAD commands.

APPENDIX A

FUNCTION FILE

The function file contains the definition of the function shift character and each function character that has been defined. It is a normal text file in the following format.

Line 1 - the first character of line 1 is the function shift character. The recommended character is the caret. The line is read as an A1 format. The caret character "^" is the default function shift character.

The remaining entries occur in pairs of lines. The first line of the pair contains the character which is to be treated as a function and a numeric integer count of how many characters of the next line are included in the function reference. This line is read in free format. The second line of the pair contains the string of zero or more characters that are the function reference value. Two lines are used for each function to be defined.

Sample Function File

```
^
#    3
ABC
$   14
THIS IS A LINE
@    4
XX
```

Note: The @ character is defined as a 4 character string ending with 2 blanks (ie, "XX ").

APPENDIX B

MACRO FILE

The macro file contains information about the PREAD command character, the initial bootstrap macro and the definition of each macro.

An optional entry to redefine the PREAD command character from its internal default of exclamation point ! to another character may be entered as the first line of the macro file. The format is the characters CC in Columns l-2, a space in Column 3 and the new PREAD command character in Column 4.

A second optional entry to define a `BOOTSTRAP` macro may be entered at the beginning of the macro file. If used, it must immediately follow the CC line, or if no CC line is used it must be the first line of the file. The characters `BOOTSTRAP` must appear beginning in Column 1 and are case sensitive. If present, the line immediately following `BOOTSTRAP` will be taken as a command line to PREAD and be executed by PREAD when a program first begins execution. Note that only the first line will be executed and if additional lines follow they will be ignored. Most typically this line would be a `RUN` command, such as, `!RUN BEGIN`. Macro `BEGIN` would be executed which may contain several other commands.

The remaining lines of the macro file are macro definitions.

Macro Syntax:

```
MACRO  name [parameter 1, parameter 2, ...]
. . . .
. . . .
. . . .
. . . .
ENDMACRO
```

Each definition begins with the line `MACRO` name where `MACRO` must begin in column 1 and the 1 to 8 character name must begin in column 7. Subsequent lines are lines to be executed when the macro is run. The macro definition is terminated by an `ENDMACRO` line beginning in column. 1. The macro name line may optionally contain one or more parameter strings. If a parameter string is used, each occurrence of the string will be replaced in the macro definition by the parameter used at execution time. The examples below illustrate the use of parameters passed to macros.

The keywords MACRO, ENDMACRO, and the macro name are all case insensitive. Old versions of software may require these to be upper case.

B.1 Sample Macro File with BOOTSTRAP

```
BOOTSTRAP
!run begin

MACRO begin
!-ECHO
TEST
A B C
endmacro

macro aBC
INPUT
25 29 59 10.31
endmacro
```

B.2 Sample Macro with Parameters

1. Macro execution line:

```
!Run Plotit RED STAGE FORECASTED
```

2. Macro definition:

```
Macro PLOTIT $LOCATION $PARAMETER $VERSION
PATH /ARKANSAS/$LOCATION/$PARAMETER/1DAY/01JAN1988/$VERSION/
PATH /ARKANSAS/$LOCATION/$PARAMETER/1DAY/01JAN1988/OBS/
Endmacro
```

3. Macro effect:

```
PATH /ARKANSAS/RED/STAGE/1DAY/01JAN1988/FORECASTED/
PATH /ARKANSAS/RED/STAGE/1DAY/01JAN1988/OBS/
```

APPENDIX C

SCREEN FILE

The screen file defines one or more screens that may be invoked by a program. A screen is composed of the material to be written to the users screen to describe the options available, a list of valid responses and the corresponding operations to perform to accomplish the response. Some additional control information describes the size of the areas, whether or not to clear the screen before and/or after displaying the screen (see action flag below) , where to position the cursor after the screen is displayed, and where to write error messages.

Elements of the screen definition are contained on lines beginning with a '#' sign. Any number of screens may be defined in the same file. The column positions of the entries defining screens must be strictly adhered to. The keywords following the # are case insensitive. Screen names are case insensitive.

The width of screens is limited to 80 columns.

Explicit function references may be placed anywhere in the text portion of the screen. The function value is substituted immediately before the screen is written. Certain control values may also be represented by function references.

C.1 - Screen definition line

Example:

```
|---5----|---15----|---25----|---35----|---45----|---55----|-

        Namexxxx Sc Sl Dl Dc P A Tl Tc E Tim V Al Ai M

#SCREEN ORD      38 18 03 23 1 1 11 50 5 060 5 07 02 2
```

The first line of a screen must be a screen definition line containing;

a) '#SCREEN' (columns 1-7),

b) one to eight character screen name (columns 9-16) If you wish to save the screen window characters and attributes to a PUF file, make certain that the screen name has a "|" as its last character. This option is only valid on the PC and is of value because subsequent reuse of the screen will be restored directly from memory instead of being put out a line at a time. For additional information, see SCREEN command "D" option.

c) Sc - number of columns required to define screen text area (columns 18-19),

d) Sl - number of lines required to define screen text area (columns 21-22),

* e) Dl - physical display line to write screen to (columns 24-25),

* f) Dc - physical display column to write screen to (columns 27-28),

* g) P - priority of screen for retention in buffer (columns 30),

* h) A - action flag, may be hexadecimal sum of items below (column 32)

1=	Before	-Force ANSI, clear screen
2=	After	-Force ANSI, clear screen
4=	Before	-Erase Graphics, Force ANSI, Clear screen
8=	Before	-Force ANSI(ANSI mode)
	After	-Force TEK

NOTE: Because this value may be a hexadecimal number, if a function character is used for this field avoid use of characters A through F as function character here.

i) Tl - total lines in TRANSLATE area (columns 33-35),

j) Tc - maximum columns in TRANSLATE area (columns 37-38).
If this value is zero, screen will be displayed but no response will be read.

* k) E - maximum sequential errors permitted before invoking response #SEQERR. If zero, no limit used. (column 40)

* l) Tim - inactivity time limit before invoking response #INACT (columns 42-44). If zero, no limit used.

* m) V - visibility level of screen (column 46). If zero, screen always visible.
(This feature not currently implemented.)

n) Al - number of lines in ATTMAP area (column 48-49). Lines are taken as same length as screen lines (item c) above.

* o) Ai - column offset index in ATTMAP line for current attributes (columns 51-52). Must point to sequence beginning with = sign. If zero, attributes will not be used.

p) M - mouse flag (column 54). If zero, no mouse areas defined; if 1, mouse areas defined with mouse inactive; if 2, mouse areas defined with mouse active. Mouse capability limited to PC screens only.

* Entry may be represented by explicit function reference. (Any non-numeric value in right position of field treated as function reference, explicit function shift character (i.e., ^) optional).

C.2 - Screen text to be displayed to user

The next lines contain an exact image of the screen that should be written to the users screen. It must be of the width and length given in C.1c and C.1d above.

Example:

```
$&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&$
$                                    $
$      Ohio River Division           $
$      &&&&&&&&&&&&&&&&&&&           $
$                                    $
$   H  -  Huntington District        $
$                                    $
$   L  -  Louisville District        $
$                                    $
$   N  -  Nashville District         $
$                                    $
$   P  -  Pittsburgh District        $
$                                    $
$                                    $
$   X  -  Exit                       $
$&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&$
$ Enter Location or 'X' to Exit:     $
$((((((((((((((((((((((((((((((((((((($
```

C.3 - Screen attributes to be displayed under text (Optional)

The #ATTRIBUTES defines the beginning of the text attribute area. The next lines contain an exact image of the screen attributes that should be written to the users screen. It must be of identical size and layout as the screen text above. Characters used in this layout area that are not given a definition in the ATTMAP area are ignored.

Example:

```
#ATTRIBUTES
$&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&$
$                                    $
$      HHHHHHHHHHHHHHHHHHH           $
$      &&&&&&&&&&&&&&&&&&&           $
$                                    $
$  SSS   IIIIIIIIIIIIIIIIIIIII       $
$                                    $
$  SSS   IIIIIIIIIIIIIIIIIIIII       $
$                                    $
$  SSS   IIIIIIIIIIIIIIIIIIIII       $
$                                    $
$  SSS   IIIIIIIIIIIIIIIIIIIII       $
$                                    $
$                                    $
$  SSS   IIIIII                      $
$&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&$
$                                    $
$((((((((((((((((((((((((((((((((((((($
```

C.4 - Attribute map area (ATTMAP)

This area defines the meaning of each character used in the attribute layout above. Any character may be used to define an attribute, even the space character. The meaning of that character may be multi-defined. The definition to be used is pointed to by the screen definition line columns 51-52. This allows attributes for monochrome, and several versions of color to be available for any screen. The attribute definition uses ANSI attribute parameters.

ANSI Parameter	Meaning
00	All attributes off
01	Bold on
04	Underscore on
05	Blink on
07	Reverse video on

Foreground	Background	Color
30	40	Black
31	41	Red
32	42	Green
33	43	Yellow
34	44	Blue
35	45	Magenta
36	46	Cyan
37	47	White

Example:

```
#ATTMAP
 =00+44+37+1         =00
H=00+44+37           =00+01
I=00+44+37+1         =00+01
S=00+47+30           =00+07
P=00+41+30           =00+07
N=00+44+37+1         =00
Z=00                 =00
```

C.5 - Prompt location

After the screen is displayed to the user the cursor will be moved to the screen location defined by the `#PROMPT` line (relative to base location of the screen). If attributes are to be used, the attributes for the prompt area and the normal attribute after a screen is exited, follows the prompt position. If a mouse is to be used, its initial position may be given following the normal attribute.

a) '#PROMPT' (column 1-7),

b) row number of prompt cursor (columns 9-10).

c) column number of prompt cursor (columns 12-13).

d) attribute prompt character (column 15),

e) normal attribute character set after leaving screen (column 17),

f) row number of mouse cursor (columns 19-20),

g) column number of mouse cursor (columns 22-23).

Example:

```
#PROMPT 17,34 N Z 05,05
```

C.6 - Message location

If an invalid response is chosen by the user, the error response will be written to the screen location defined (relative to base location of the screen). If attributes are to be used, the message attribute follows the message location.

a) '#MESSAGE' (column 1-8),

b) row number of message location (columns 10-11).

c) column number of message location (columns 13-14).

d) message attribute character (column 16),

Example:

```
#MESSAGE 14,05 P
```

C.7 - Response list and translation area.

The response list and translation area contains a list of all valid responses, and the action to be taken when a response is entered.

The first line of the TRANSLATE area must contain:

a) '#TRANSLATE' (column 1-10),

b) number of total lines in TRANSLATE area (columns 11-13).
(Total lines may now exceed previous limit of 99).

c) optional type-ahead buffer flush indicator, FLUSH (columns 15-19)
If the optional FLUSH parameter appears on this line the system type-ahead buffer will be flushed before accepting response from this screen. This may be used on error recovery, or other critical screens where type-ahead may not be appropriate.

The remaining lines define valid responses:

c) number of lines executed for this response (column 1), must be in range 1 to 9,

d) response control character (column 2), control how rigid response must be,

if	' '	>	abbreviation permitted, case sensitive,
if	':'	>	abbreviation prohibited, case sensitive, (must be exact)
if	','	>	abbreviation permitted, case insensitive, (most relaxed)
if	';'	>	abbreviation prohibited, case insensitive.

e) 'response' one to eight characters (column 3-10),

f) action to 'response' (column 12-mm), where mm= is maximum column given in C.1j

g) optional, if col 54 of `#SCREEN` line is 1 or 2, a rectangular hit area is specified for mouse pick to be equivalent to keyboard response. Format is [r1,c1,r2,c2] where:
r1,c1 define upper left corner of hit box, and
r2,c2 define lower right corner of hit box.

If a response of `#ANY` appears as the last entry in the translate area, it will always be used if no previous keyboard responses match.

If control characters are desired as valid responses they may be represented in the translate area by three character sequences of ^nm, where the nm is the ANSI 2-character representation of the control character. (e.g., `^CR, ^BL,^EC`).

Example:

```
#TRANSLATE 11
1;H        !RUN ORD H     [06,03,06,38]
1;L        !RUN ORD L     [08,03,08,38]
1;N        !RUN ORD N     [10,03,10,38]
1;P        !RUN ORD P     [12,03,12,38]
1,X        !RUN FINISH    [15,03,15,13]
1;EXIT     !RUN FINISH
1:^CR      !RUN FINISH
1:XxYy?    !SC ORD        [01,01,18,38]
1:#SEQERR  !RUN ERRORS
1:#INACT   !RUN TIMEOUT
1;OP       !SC  OPTION1|
#ENDSCREEN
```

C.8 - End of Screen

The end of the screen is defined by '`#ENDSCREEN`' (columns 1-10).

C.9 - Full Example Screen

```
|---5----|---15----|---25----|---35----|---45----|---55----|-

        Namexxxx Sc Sl Dl Dc P A Tl Tc E Tim V Al Ai M

#SCREEN ORD      38 18 03 23 1 1 11 50 5 060 5 07 02 2
$&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&$
$                                    $
$      Ohio River Division           $
$      &&&&&&&&&&&&&&&&&&&           $
$                                    $
$   H  -  Huntington District        $
$                                    $
$   L  -  Louisville District        $
$                                    $
$   N  -  Nashville District         $
$                                    $
$   P  -  Pittsburgh District        $
$                                    $
$                                    $
$   X  -  Exit                       $
$&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&$
$ Enter Location or 'X' to Exit:     $
$((((((((((((((((((((((((((((((((((((($
#ATTRIBUTES
$&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&$
$                                    $
$      HHHHHHHHHHHHHHHHHHH           $
$      &&&&&&&&&&&&&&&&&&&           $
$                                    $
$  SSS   IIIIIIIIIIIIIIIIIIII        $
$                                    $
$  SSS   IIIIIIIIIIIIIIIIIIII        $
$                                    $
$  SSS   IIIIIIIIIIIIIIIIIIII        $
$                                    $
$  SSS   IIIIIIIIIIIIIIIIIIII        $
$                                    $
$                                    $
$  SSS   IIIIII                      $
$&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&$
$                                    $
$((((((((((((((((((((((((((((((((((((($
#ATTMAP
 =00+44+37+01        =00
H=00+44+37           =00+01
I=00+44+37+01        =00+01
S=00+47+30           =00+07
P=00+41+30           =00+07
N=00+44+37+01        =00
Z=00                 =00
#PROMPT 17,34 N Z 05,05
#MESSAGE 14,05 P
#TRANSLATE 11
1;H        !RUN ORD H     [06,03,06,38]
1;L        !RUN ORD L     [08,03,08,38]
1;N        !RUN ORD N     [10,03,10,38]
1;P        !RUN ORD P     [12,03,12,38]
1,X        !RUN FINISH    [15,03,15,13]
1;EXIT     !RUN FINISH
1:^CR      !RUN FINISH
1:XxYy?    !SC ORD        [01,01,18,38]
1:#SEQERR  !RUN ERRORS
1:#INACT   !RUN TIMEOUT
1;OP       !SC  OPTION1|
#ENDSCREEN
```

APPENDIX D

SAMPLES

D.1 Sample macro file

```
BOOTSTRAP
!RUN BEGIN

---------------------------------------------------------------------------

MACRO BEGIN
!IF ( '^K' .EQ. 'COPY-IT' ) THEN
!JCL COPY REPORT ^F >NUL
!TEACH K NULL
!PAGE
!RUN REPVUE ^B ^T CHOICES
!SC ^D
!ENDIF
!SC MAIN
FIN
ENDMACRO

--------------------------------------------------------------------------

MACRO REPORT $DIST
!TEACH D $DIST
!IF ( '^D' .EQ. 'ORP' ) THEN
!TEACH A  P
!SC ORP
!ELSEIF ( '^D' .EQ. 'ORH' ) THEN
!TEACH A  H
!SC ORH
!ELSEIF ( '^D' .EQ. 'ORL' ) THEN
!TEACH A  L
!SC ORL
!ELSE
!SC ORD
!ENDIF
!SC MAIN
ENDMACRO

---------------------------------------------------------------------------

MACRO SITE $L $NAME $ID $T $NXT
!IF ( '$L' .NE. '---' ) THEN
!TEACH L '$L'
!TEACH N '$NAME'
!ENDIF
!IF ( '^L' .NE. '---' ) THEN
!RUN TYPE-^T
!SC CHOICES
!ENDIF
!SC $NXT
ENDMACRO
```

D.2 Example macro file from IA5/MOD5/HEC5 workshop

```
BOOTSTRAP
!RUN BEGIN

MACRO BEGIN
!RUN SAVEPO
!-FUNCTION
^z
TIME 0100 20JAN59 2400 23JAN59
GR,7
!* SH,ON
!-ECHO
!RUN MAP
FINISH
ENDMACRO

MACRO WC
US Worthington (CP #268)
PA  /SCIOTO/WOOE2/FLOW-CHAN CAP/01JAN1959/3HOUR/^r^s^p5^o/
PA  /SCIOTO/WOOE2/FLOW-REG/01JAN1959/3HOUR/^r^s^p5^o/
PA  /SCIOTO/WOOE2/PRECIP-INC/01JAN1959/3HOUR/^r^s^p/
PA  /SCIOTO/WOOE2/FLOW-LOC CUM/01JAN1959/3HOUR/^r^s^p5^o/
PA  /SCIOTO/WOOE2/FLOW-FLOOD RES/01JAN1959/3HOUR/^r^s^p5^o/
TY 1 1 2 1 1
SW BLN TRI
SP,75
PLOT.A
!SC OPTIONS
ENDMACRO

MACRO CC
US Scioto River at Columbus (CP #275)
PA  /SCIOTO/CLSF4/FLOW-CHAN CAP/01JAN1959/3HOUR/^r^s^p5^o/
PA  /SCIOTO/CLSF4/FLOW-REG/01JAN1959/3HOUR/^r^s^p5^o/
PA  /SCIOTO/CLSF4/PRECIP-INC/01JAN1959/3HOUR/^r^s^p/
PA  /SCIOTO/CLSF4/FLOW-LOC CUM/01JAN1959/3HOUR/^r^s^p5^o/
PA  /SCIOTO/CLSF4/FLOW-FLOOD RES/01JAN1959/3HOUR/^r^s^p5^o/
TY 1 1 2 1 1
SW BLN TRI
SP,75
PLOT.A
!SC OPTIONS
ENDMACRO
```

D.3 Example screen file MODCON/IA5/MOD5/HEC5 Workshop

```
#SCREEN MAP      80 20 04 01 1 2 20 35
 -------------  Scioto Basin Locations  ------------------

    Delaware Res (D)
         |
         |
   Worthington (W)    Alum Ck Res (A)
         |                |
         |                |
  Columbus (C)       E. Col-Alum Ck (E)
         |                |
         |                |------ Hoover Res (H)
         |                |
         |------------ Rees (R)
         |                                 Precipitation Alt =  ^p
         |                                 Operations Alt = ^o
  Circleville (CV)                         ( Enter P to change Precipitation, )
         |                                 (    or Operations Alternative     )
         |

 Select location, or X to Exit ==>
#PROMPT 20,36
#MESSAGE 19,02
#TRANSLATE 20
2 D         !RUN DR
            !RUN MAP
2 W         !RUN WC
            !RUN MAP
2 C         !RUN CC
            !RUN MAP
2:CV        !RUN CVC
            !RUN MAP
2 A         !RUN AR
            !RUN MAP
2 E         !RUN EC
            !RUN MAP
2 H         !RUN HR
            !RUN MAP
2 R         !RUN RC
            !RUN MAP
2 X         !RUN RESTORPO
            FINISH
2 P         !RUN OPTSET
            !RUN MAP
#ENDSCREEN
```

```
#SCREEN OPT      79 18 05 01 1 2 09 25
               Change Precipitation or Operations Alternative
               ----------------------------------------------

               P-A    Select Precipitation Alternative 'A'
               P-B    Select Precipitation Alternative 'B'
               P-C    Select Precipitation Alternative 'C'
               P-D    Select Precipitation Alternative 'D'

               O-A    Select Operations Alternative 'A'
               O-B    Select Operations Alternative 'B'
               O-C    Select Operations Alternative 'C'
               O-D    Select Operations Alternative 'D'

               X      Exit


               Select desired option ==>

#PROMPT 17,42
#MESSAGE 16,16
#TRANSLATE 09
1:P-A      !TEACH p A
1:P-B      !TEACH p B
1:P-C      !TEACH p C
1:P-D      !TEACH p D
1:O-A      !TEACH o A
1:O-B      !TEACH o B
1:O-C      !TEACH o C
1:O-D      !TEACH o D
1 X        !PR Exit
#ENDSCREEN
```

```
#SCREEN OPTIONS  79 03 01 01 1 9 24 25
 Enter X, option, or ? for Help
 ==>

#PROMPT 02,06
#MESSAGE 01,35
#TRANSLATE 24
2 ^CR      PLOT
           !SC OPTIONS
3 S        SHADE,ON
           PLOT
           !SC OPTIONS
3 SN       SHADE,OFF
           PLOT
           !SC OPTIONS
3 G        GRID,ON
           PLOT
           !SC OPTIONS
3 GN       GRID,OFF
           PLOT
           !SC OPTIONS
2 W        W
           !SC OPTIONS
3 HELP     !SC HELP
           PLOT
           !SC OPTIONS
3 ?        !SC HELP
           PLOT
           !SC OPTIONS
2 X
           !SC CLEAR
#ENDSCREEN
```

```
#SCREEN HELP     80 20 01 01 1 7 01 25

                             DSPLAY - Help Screen
                         -----------------------------

     After completing the plot the user may exercise several options.

         Option              Remarks
         ------              -------
          X                  Exit plot and continue
          S                  Shade
          SN                 Shade-No
          G                  Grid
          GN                 Grid-No
          W                  Window the plot area by setting cross-hairs
                             (After the cross-hairs are displayed move to
                             the desired left side and press W, then
                             move to the desired right side and press
                             W again.)

          Press <RETURN> to restore original plot.
#PROMPT 20,10
#MESSAGE 21,03
#TRANSLATE 01
1 ^CR      !PAGE GRAPHICS
#ENDSCREEN
```

D.4 Example macros using parameters

```
MACRO RIVDISP APART BPART CPART FPART DOIT NEXT1
RESET
PA /APART/BPART/FLOW/01JAN1988/1DAY/FPART/
PA /APART/BPART/STAGE/01JAN1988/1DAY/FPART/
!RUN DOIT APART BPART
!SCREEN NEXT1
ENDMACRO

MACRO RESDISP APART BPART CPART FPART DOIT NEXT1
RESET
PA /APART/BPART/STORAGE/01JAN1988/1DAY/FPART/
!RUN DOIT APART BPART
!SCREEN NEXT1
ENDMACRO

MACRO HELP NAME NEXT1
REP XINDEX=`H-DSP.X
REP NAME
!SCREEN NEXT1
ENDMACRO

MACRO SETPLOT NEXT1
!TEACH $ PLOTIT
TIME &
!SCREEN NEXT1
ENDMACRO

MACRO SETTAB NEXT1
!TEACH $ TABIT
TIME T-10D T+30D
!SCREEN NEXT1
ENDMACRO

MACRO #ABORT
!PR
!PR - - Aborting - -
!PR
!WAIT 2
ABORT
ENDMACRO
```